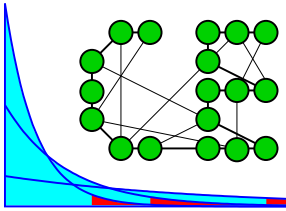
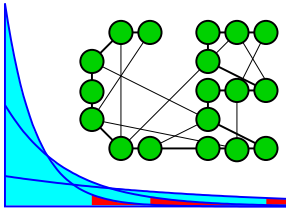


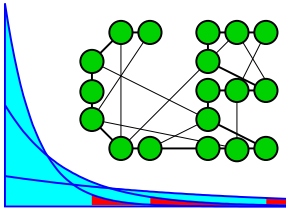
THE CROSS-ENTROPY METHOD IN OPTIMIZATION AND MONTE-CARLO SIMULATION

Reuven Y. Rubinstein

Faculty of Industrial Engineering and Management, Technion, Israel

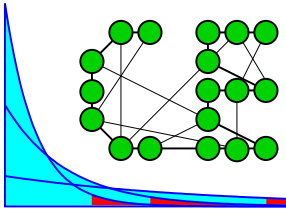


Contents



CE Matters

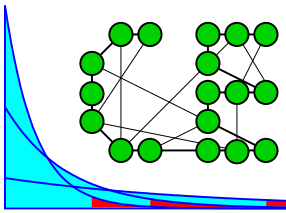
Book: R.Y. Rubinstein and D.P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*, Springer-Verlag, New York, 2004.



CE Matters

Book: R.Y. Rubinstein and D.P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*, Springer-Verlag, New York, 2004.

Special Issue: *Annals of Operations Research* (Jan 2005).



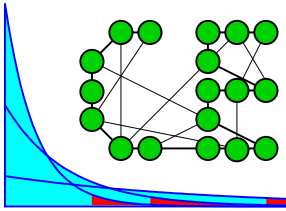
CE Matters

Book: R.Y. Rubinstein and D.P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*, Springer-Verlag, New York, 2004.

Special Issue: *Annals of Operations Research* (Jan 2005).

The CE **home page**:

`http://www.cemethod.org`



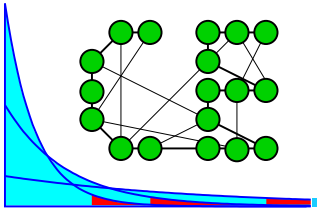
Introduction

The Cross-Entropy Method was originally developed as a simulation method for the estimation of *rare event* probabilities:

Estimate $\mathbb{P}(S(\boldsymbol{X}) \geq \gamma)$

$\boldsymbol{X}$: random vector/process taking values in some set $\mathcal{X}$.

S: function on $\mathcal{X}$.



Introduction

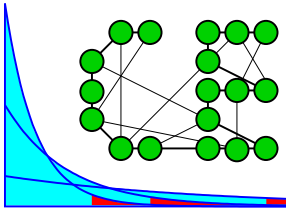
The Cross-Entropy Method was originally developed as a simulation method for the estimation of *rare event* probabilities:

Estimate $\mathbb{P}(S(\boldsymbol{X}) \geq \gamma)$

$\boldsymbol{X}$: random vector/process taking values in some set $\mathcal{X}$.
S: function on $\mathcal{X}$.

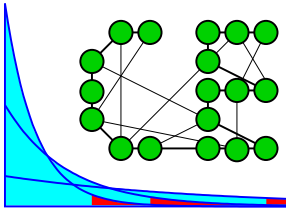
It was soon realised that the CE Method could also be used as an *optimization* method:

Determine $\max_{\boldsymbol{x} \in \mathcal{X}} S(\boldsymbol{x})$



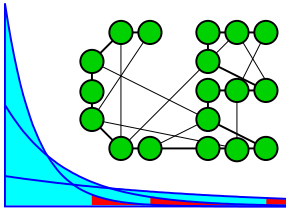
Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).



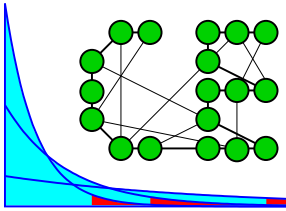
Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.



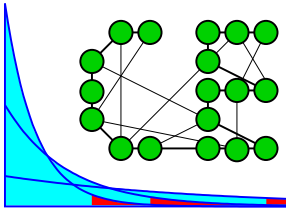
Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.
- Machine Learning



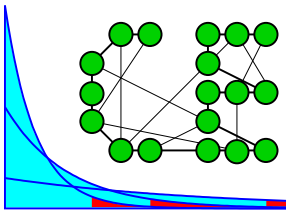
Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.
- Machine Learning
- Pattern Recognition, Clustering and Image Analysis



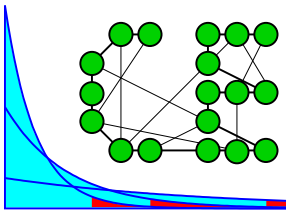
Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.
- Machine Learning
- Pattern Recognition, Clustering and Image Analysis
- DNA Sequence Alignment



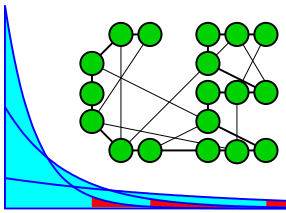
Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.
- Machine Learning
- Pattern Recognition, Clustering and Image Analysis
- DNA Sequence Alignment
- Simulation-based (noisy) Optimization, like Optimal Buffer Allocation and Optimization in Finance Engineering



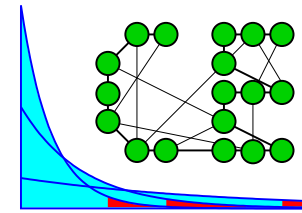
Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.
- Machine Learning
- Pattern Recognition, Clustering and Image Analysis
- DNA Sequence Alignment
- Simulation-based (noisy) Optimization, like Optimal Buffer Allocation and Optimization in Finance Engineering
- Multi-extremal Continuous Optimization

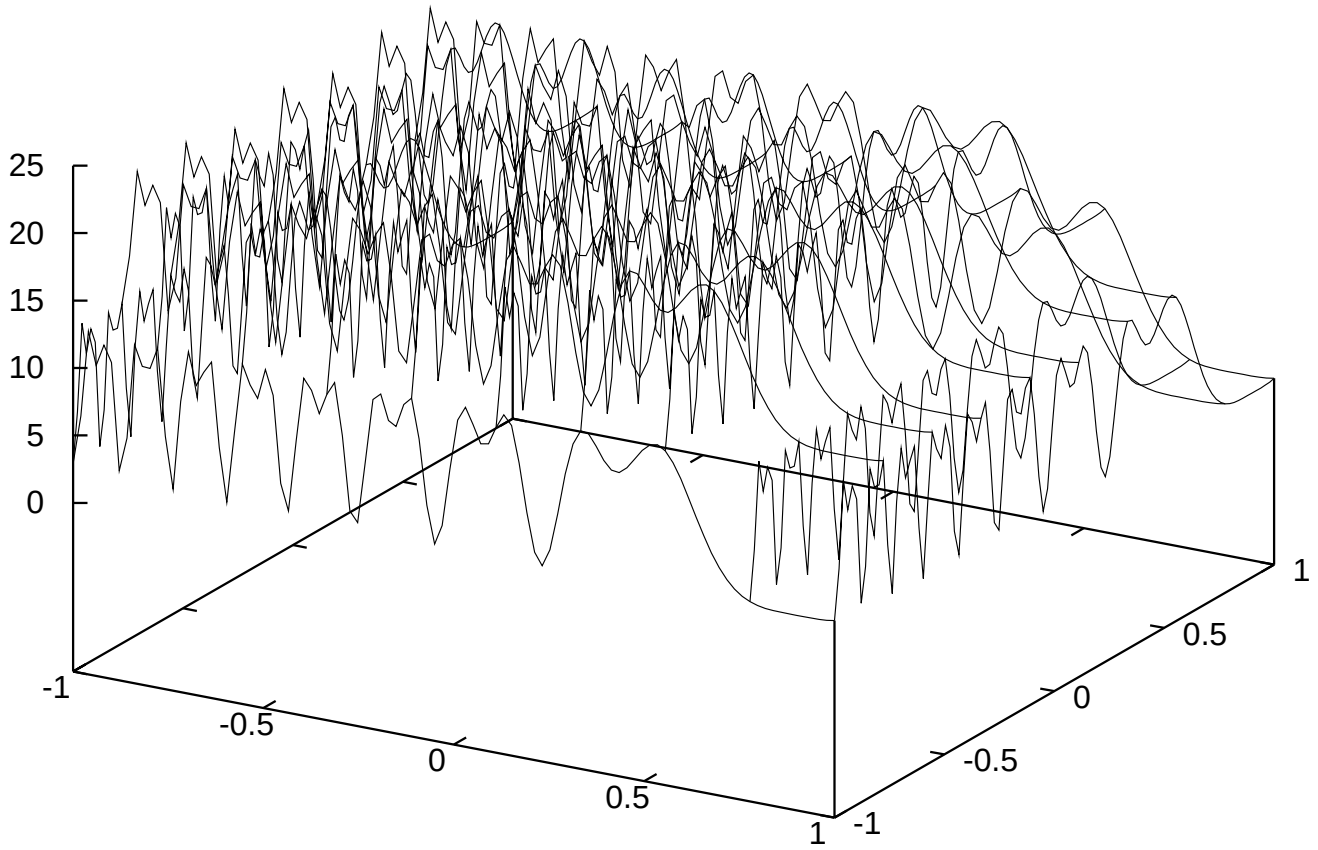


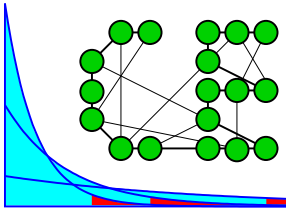
Applications

- Rare-Event Probability Estimation (Queueing Models), Estimation of Normalization Constant and Counting Problems (Permanent Calculation, Self-Avoiding Random Walks).
- Combinatorial Optimization, like TSP, Maximal Cut, Scheduling and Production Lines.
- Machine Learning
- Pattern Recognition, Clustering and Image Analysis
- DNA Sequence Alignment
- Simulation-based (noisy) Optimization, like Optimal Buffer Allocation and Optimization in Finance Engineering
- Multi-extremal Continuous Optimization



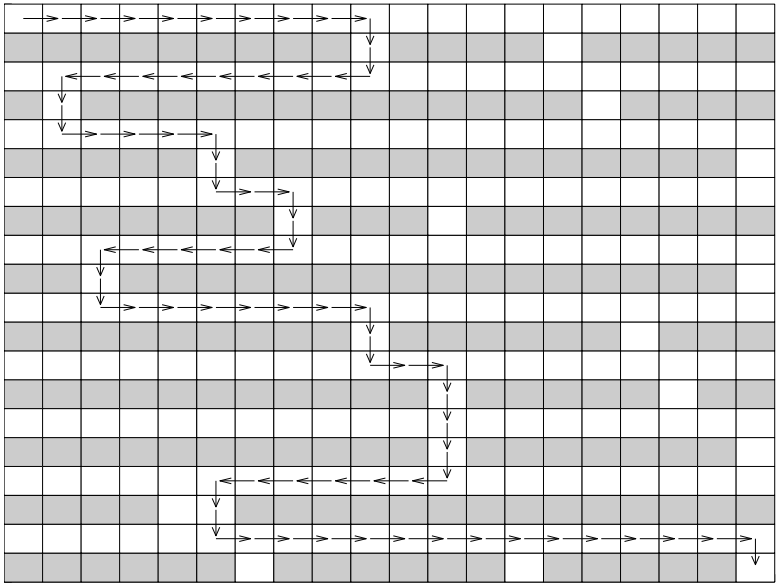
A Multi-extremal function

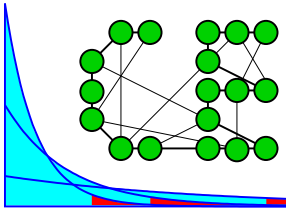




A Maze Problem

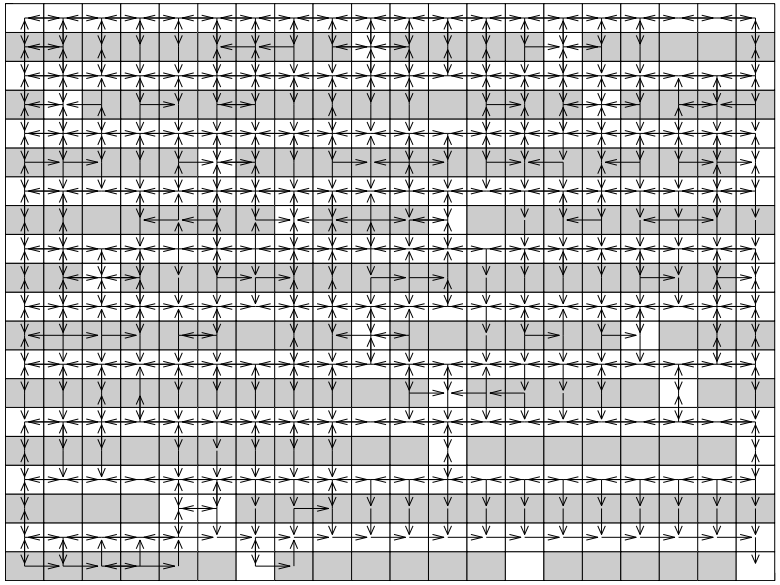
The Optimal Trajectory

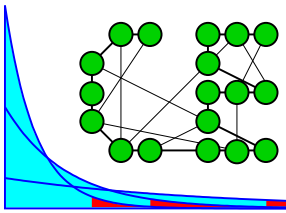




A Maze Problem

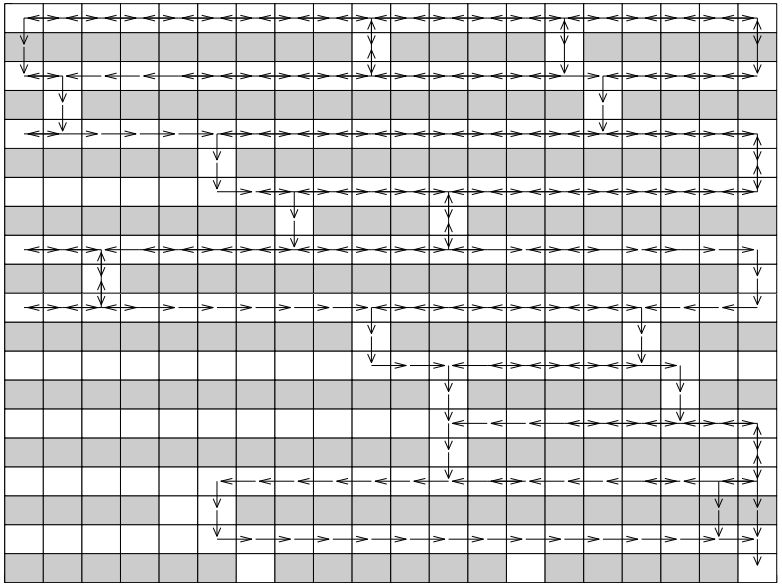
Iteration 1:

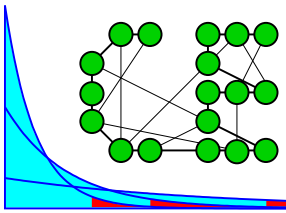




A Maze Problem

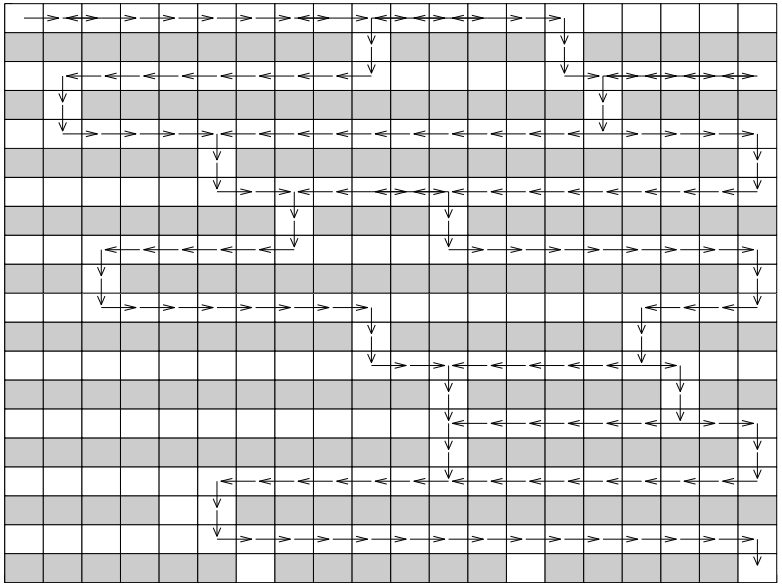
Iteration 2:

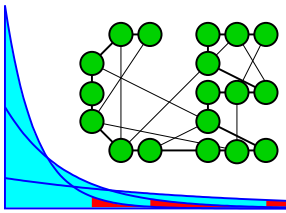




A Maze Problem

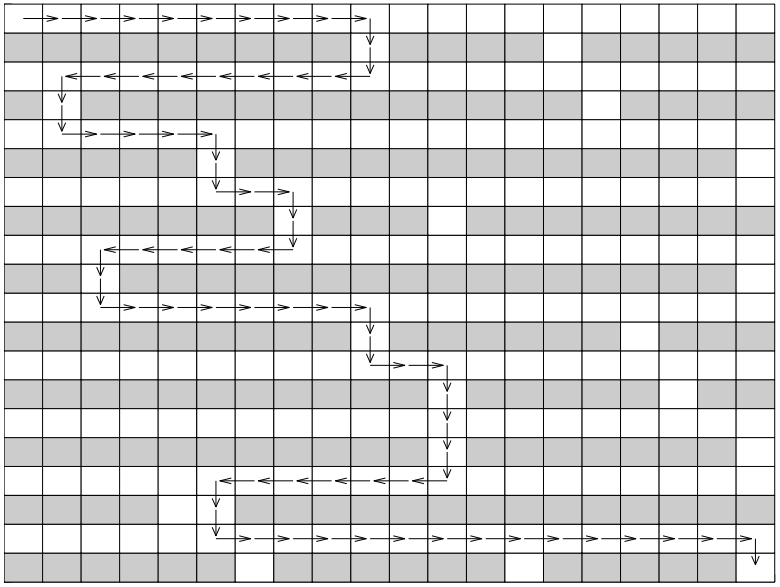
Iteration 3:

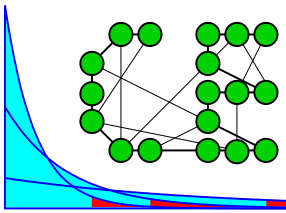




A Maze Problem

Iteration 4:





Calculating Number of Trajectories in a Partially Connected TSP

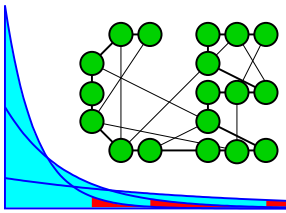
A Motivating Example

Consider the following distance matrix C of partially connected TSP with $n = 4$.

$$C = \begin{pmatrix} 0 & c_{12} & c_{13} & c_{14} \\ c_{21} & 0 & c_{23} & c_{13} \\ \infty & c_{32} & 0 & c_{34} \\ c_{41} & \infty & c_{43} & 0 \end{pmatrix}. \tag{1}$$

The associated 0-1 distance matrix D is

$$D = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \tag{2}$$



Partially Connected TSP

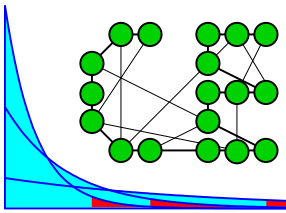
Valid and Not Valid Trajectories

Define a valid TSP trajectory as one which visits each city only once and goes trough all n cities with $d_{ij} = 1$. In our example with

$$D = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

we have a total of $(n-1)! = 3! = 6$ trajectories, three of wich, $1 \to 2 \to 3 \to 4 \to 1$, $1 \to 3 \to 2 \to 4 \to 1$ and $1 \to 4 \to 3 \to 2 \to 1$ are valid and the remaining 3 are invalid.

Denote $|\mathcal{X}| = (n-1)!$ for a fully connected TSP. Let $\mathcal{X}^-$ be a subset of $|\mathcal{X}|$ and let $|\mathcal{X}^-|$ be the desired cordinality of $\mathcal{X}^-$, respectively. We have $|\mathcal{X}| = 3! = 6$ and $|\mathcal{X}^-| = 3$, respectively.



Partially Connected TSP

Trajectory Generation

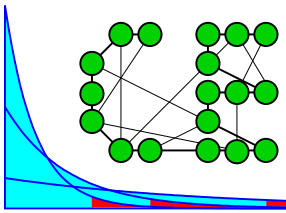
We shall associate with each cost matrix D a probability matrix. For our D matrix

$$D = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

we can take, as example, the following initial one

$$P_0 = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 1/3 & 1/3 & 0 \end{pmatrix}. \tag{3}$$

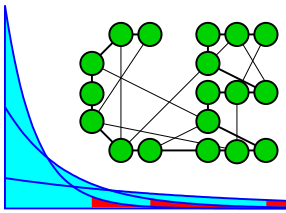
We shall call such TSP problem the 0-1-TSP.



Partially Connected TSP

Trajectory Generation in a TSP

1. Define $P^{(1)} = P$ and $X_1 = 1$. Let $k = 1$.
2. Obtain $P^{(k+1)}$ from $P^{(k)}$ by first setting the X_k-th column of $P^{(k)}$ to 0 and then normalizing the rows to sum up to 1. Generate X_{k+1} from the distribution formed by the X_k-th row of $P^{(k)}$.
3. If $k = n - 1$ then **stop**; otherwise set $k = k + 1$ and reiterate from step 2.



Partially Connected TSP

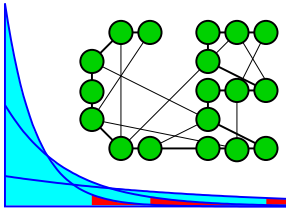
The Rare Event Approach

Define

$$\ell = \mathbb{E}_{\mathcal{X}} I_{\{\boldsymbol{X} \in A\}}, \tag{4}$$

where $|\mathcal{X}| = (n-1)!$ and A denotes the event that $\boldsymbol{X}$ is a valid TSP trajectory in the sample space $\mathcal{X}$ and $\mathcal{X}$ in $\mathbb{E}_{\mathcal{X}}$ means that the expectation is taken with respect to the uniform pdf $f(\boldsymbol{x})$ over $\mathcal{X}$, that is

$$f(\boldsymbol{x}) = \frac{1}{|\mathcal{X}|} = \frac{1}{n-1!}. \tag{5}$$



Partially Connected TSP

The Rare Event Approach

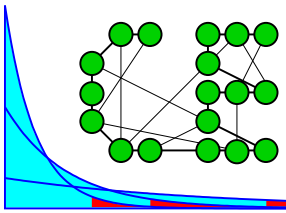
Clearly, that

$$|\mathcal{X}^-| = \ell |\mathcal{X}|.$$

It is obvious that the naive Monte-Carlo estimates of ℓ, that is

$$\widetilde{\ell} = \frac{1}{N} \sum_{i=1}^{N} I_{\{\boldsymbol{X}_i \in A\}} \tag{6}$$

in $|\widetilde{\mathcal{X}}^-| = \widetilde{\ell} |\mathcal{X}|$ is useless.



Partially Connected TSP

The Rare Event Approach: Importance Sampling

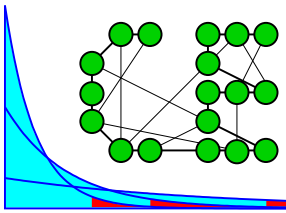
To speed up the simulation process we use *importance sampling* (IS). In particular, we write ℓ as

$$\ell = \mathbb{E}_{\mathcal{X}}\{I_{\{\boldsymbol{X}\in A\}} W(\boldsymbol{X})\}, \tag{7}$$

where $\mathcal{X}$ in $\mathbb{E}_{\mathcal{X}}$ means that the random trajectory $\boldsymbol{X}$ is generated according to the above TSP generation Algorithm.

$$W(\boldsymbol{x}) = \frac{f(\boldsymbol{x}, \boldsymbol{P}_0)}{f(\boldsymbol{x}, \boldsymbol{P})}$$

is the likelihood ratio with $f(\boldsymbol{x})$ being uniformly distributed over $\mathcal{X}$, and $f(\boldsymbol{x}, \boldsymbol{P})$ being the IS pdf.



Partially Connected TSP

The Rare Event Approach

The IS estimates of ℓ in and $|\mathcal{X}^-|$ are

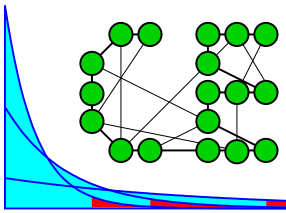
$$\hat{\ell} = \frac{1}{N}\sum_{i=1}^{N} I_{\{\boldsymbol{X}_i \in A\}} W(\boldsymbol{X}_i),\ W(\boldsymbol{X}_i) = \frac{f(\boldsymbol{X}_i, \boldsymbol{P}_0)}{f(\boldsymbol{X}_i, \boldsymbol{P})} \qquad (8)$$

and

$$|\widehat{\mathcal{X}}^-| = \widehat{\ell}|\mathcal{X}| = \frac{1}{N}\sum_{i=1}^{N} I_{\{\boldsymbol{X}_i \in A\}} \frac{1}{f(\boldsymbol{X}_i, \boldsymbol{P})}, \qquad (9)$$

respectively. The second equality holds since $f(\boldsymbol{x})$ is uniformly distributed over the set $\mathcal{X}$, that is

$$f(\boldsymbol{x}, \boldsymbol{P}_0) = f_1(x_1) f_2(x_2|x_1) \cdots f_n(x_n|x_1, \ldots, x_{n-1}) = \frac{1}{n-1}\frac{1}{n-2}\cdots$$



Partially Connected TSP

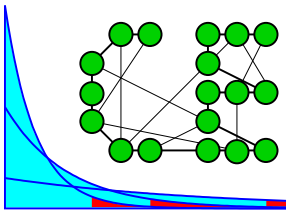
The Rare Event Approach

We shall use interchangeable both events $\{\boldsymbol{X} \in A\}$ and $\{S(\boldsymbol{X}) = n\}$ in (8) and (9), since

$$\{\boldsymbol{X} \in A\} \equiv \{S(\boldsymbol{X}) = n\}. \tag{10}$$

Here $S(\boldsymbol{X})$ is the length of the 0-1-TSP trajectory. Clearly, that $0 \leq S(\boldsymbol{X}) \leq n$ and for a valid trajectory, that is for $\{\boldsymbol{X}_i \in A\}$, we have that $S(\boldsymbol{X}) = n$. The optimal in the cross-entropy sense matrix $\boldsymbol{P}$ is

$$\boldsymbol{P}^* = \underset{\boldsymbol{P}}{\operatorname{argmax}}\ \mathbb{E}_{\boldsymbol{P}_0}\, I_{\{S(\boldsymbol{X}) \geq \gamma\}} \log f(\boldsymbol{X}; \boldsymbol{P}). \tag{11}$$



Partially Connected TSP

Example

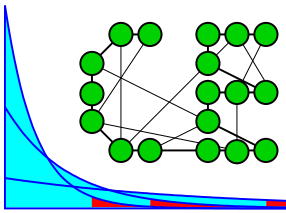
For our example with

$$D = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

and any initial P_0, the optimal P^* is

$$P^* = \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 \end{pmatrix}. \qquad (12)$$

For example, for the trajectory $1 \to 3 \to 2 \to 4 \to 1$ we have $f(\boldsymbol{x}, \boldsymbol{P}) = 1/3 \cdot 1 \cdot 1 = 1/3$.

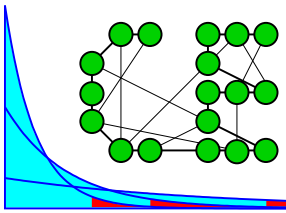


Partially Connected TSP

Updating P and γ

Adaptive updating of γ_t. Let γ_t be the $(1-\rho)$-quantile of $S(\boldsymbol{X})$ under $\boldsymbol{P}_{t-1}$. A simple estimator, denoted $\widehat{\gamma}_t$, of γ_t can be obtained by drawing a random sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from $f(\boldsymbol{x}, \boldsymbol{P}_{t-1})$, calculating the associated function values $S(\boldsymbol{X}_1), \ldots, S(\boldsymbol{X}_N)$ and their order statistics $S_{(1)}, \ldots, S_{(N)}$ and assigning $\widehat{\gamma}_t$ to be these order statistics' $(1-\rho)$-quantile, that is,

$$\widehat{\gamma}_t = S_{(1-\rho)N+1}. \tag{13}$$



Partially Connected TSP

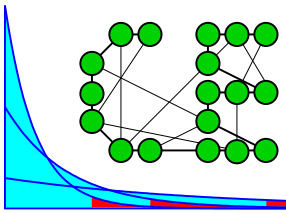
Updating P and γ

Adaptive updating of $\boldsymbol{P}_t$. For fixed γ_t and $\boldsymbol{P}_{t-1}$, derive $\boldsymbol{P}_t$ from the solution of the program

$$\max_{\boldsymbol{P}_t} D(\boldsymbol{P}_t) = \max_{\boldsymbol{P}_t} \mathbb{E}_{\boldsymbol{P}_{t-1}} I_{\{S(\boldsymbol{X}) \geq \gamma_t\}} \log f(\boldsymbol{x}, \boldsymbol{P}_t) \; . \qquad (14)$$

The stochastic counterpart of (14) is as follows: for fixed $\widehat{\gamma}_t$ and $\widetilde{\boldsymbol{P}}_{t-1}$, derive $\widetilde{\boldsymbol{P}}_t$ from the following program

$$\max_{\widetilde{\boldsymbol{P}}_t} \widehat{D}(\widetilde{\boldsymbol{P}}_t) = \max_{\widetilde{\boldsymbol{P}}_t} \frac{1}{N} \sum_{i=1}^{N} I_{\{S(\boldsymbol{X}_i) \geq \widehat{\gamma}_t\}} \log f(\boldsymbol{X}_i; \widetilde{\boldsymbol{P}}_t) \; . \qquad (15)$$



Partially Connected TSP

The Rare Event Frame

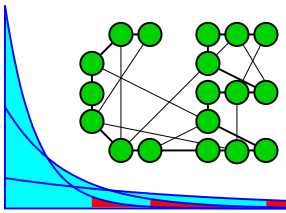
Solving(14) and (15) for the TSP problem we obtain the following simple analytic expressions

$$p_{t,ij} = \frac{\mathbb{E}_{\boldsymbol{P}_{t-1}} I_{\{\boldsymbol{X} \in \mathcal{X}_{ij}\}} I_{\{S(\boldsymbol{X}) \geq \gamma_t\}}}{\mathbb{E}_{\boldsymbol{P}_{t-1}} I_{\{S(\boldsymbol{X}) \geq \gamma_t\}}} \tag{16}$$

and

$$\widetilde{p}_{t,ij} = \frac{\sum_{k=1}^{N} I_{\{\boldsymbol{X}_k \in \mathcal{X}_{ij}\}} \, I_{\{S(\boldsymbol{X}_k) \leq \widehat{\gamma}_t\}}}{\sum_{k=1}^{N} I_{\{S(\boldsymbol{X}_k) \leq \widehat{\gamma}_t\}}} \tag{17}$$

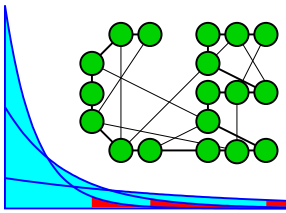
for updating the parameters of the matrices $\boldsymbol{P}_t$ and $\widetilde{\boldsymbol{P}}_t$.



The Main Algorithm

The 0-1-TSP Algorithm

1. Choose some $\widehat{\boldsymbol{P}}_0$, say $p_{0,ij} = \frac{1}{n-1}$. Set $t = 1$.
2. Generate a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ from the density $f(\cdot; \widehat{\boldsymbol{P}}_{t-1})$ and compute the sample $(1-\rho)$-quantile $\widehat{\gamma}_t$ of the performances according to $\widehat{\gamma}_t = S_{(1-\rho)N+1}$.
3. Use the **same** sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ and solve the stochastic program (15). Denote the solution by $\widehat{\boldsymbol{P}}_t$.
4. Apply smooting $\widehat{\boldsymbol{P}}_t = \alpha \widetilde{\boldsymbol{P}}_t + (1-\alpha)\widetilde{\boldsymbol{P}}_{t-1},\ 0 < \alpha < 1$.
5. If $\hat{\gamma}_t < n$, set $t = t+1$ and reiterate from step 2. Else proceed with step 6.
6. Estimate $|\mathcal{X}^-|$ according to (9).



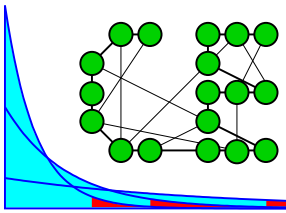
Calculating the Permanent and More

The permanent problem can be stated as follows: Given an n x n matrix $A = (a_{ij})$, where each element a_{ij} is either 0 or 1, and a set of all permutations Π of the sequence $\{1, \ldots, n\}$, find the permanent of A, which is

$$\mathrm{perm}(A) = \sum_{\sigma \in \Pi} \prod_{i=1}^{n} a_{i\sigma(i)}.$$

Here $\sigma \in \Pi$ is a permutation of $\{1, \ldots, n\}$, which can be represented as

$$\begin{pmatrix} 1, & 2, & \ldots, & n \\ \sigma(1), & \sigma(2), & \ldots, & \sigma(n) \end{pmatrix}, \qquad (18)$$



Calculating the Permanent

$$\mathcal{R}_A = \{\sigma : \ S(\sigma) = 1\}, \ S(\sigma) = \prod_{i=1}^{n} a_{i\sigma(i)}, \qquad (19)$$

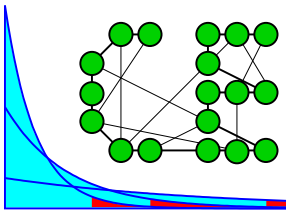
where $\sigma(i)$ records the new position of label i after the permutation.

Note that if one of the elements $a_{i\sigma(i)}, \ i = 1, \ldots, n$ is zero, then $S(\sigma)$ is zero.

A 3 x 3 permanent matrix

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

For the matrix A there are $3! = 6$ permutations. They are:



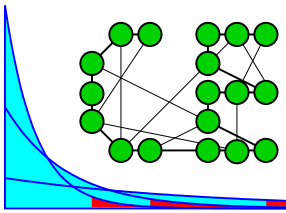
Permanent Example

$$\sigma = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \\ 3 & 2 & 1 \end{pmatrix}, \tag{20}$$

with the corresponding $S(\sigma)$ values

$$S(\sigma) = \begin{pmatrix} a_{11}\, a_{22}\, a_{33} & = 1 \\ a_{11}\, a_{23}\, a_{32} & = 0 \\ a_{12}\, a_{21}\, a_{33} & = 1 \\ a_{12}\, a_{23}\, a_{31} & = 0 \\ a_{13}\, a_{21}\, a_{32} & = 1 \\ a_{13}\, a_{22}\, a_{33} & = 0 \end{pmatrix}, \tag{21}$$

and the permanent $\mathrm{perm}(A) = 3$.



Calculating the Permanent

A naive way for approximating $\mathrm{perm}(A)$ is to generate the permutations uniformly on the set Π and then to estimate it as

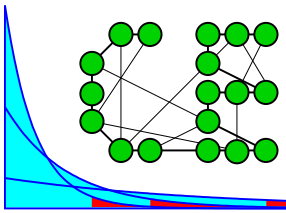
$$\widehat{\mathrm{perm}}(A) = \frac{\sum_{j=1}^{N} S(\sigma_j)}{N} n! \tag{22}$$

However, for sparse matrix A we typically have $\mathrm{perm}(A) << n!$, that is under the uniform sampling over Π, the probability

$$\ell = P\{S(\sigma) = 1\} = \mathbb{E}_U\{I_{\prod_{i=1}^{n} a_{i\sigma(i)}} = 1\} \tag{23}$$

is very small and $\{S(\sigma) = 1\}$ is rare event. Clearly that

$$\mathrm{perm}(A) = \ell n! \tag{24}$$



Permanent Calculation

Note that ℓ can be also written equivalently as

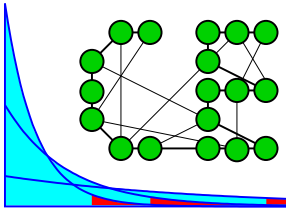
$$\ell = P\{\mathcal{S}(\sigma) = n\} = \mathbb{E}_U\{I_{\sum_{i=1}^n a_{i\sigma(i)} = n}\}, \qquad (25)$$

where

$$\mathcal{S}(\sigma) = \sum_{i=1}^{n} a_{i\sigma(i)}, \qquad (26)$$

which as more suitable for CE.

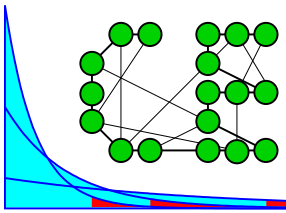
Here ℓ defines a probability that the length of a random permutation (trajectory) equals $exactly\, n$. Thus we have a $longest\ tour$ TSP-like model, where (a) the distances between and within cities are either zeros or ones; (b) TSP tour is defined as follows: as soon as all cities are visited from some starting city, one does not loop back, but rather remains there for ever.



Numerical Results

The Permanent Problem

As an illustration of the dynamics of the CE algorithm, we display below the sequence of matrices $\widehat{\boldsymbol{P}}_0, \widehat{\boldsymbol{P}}_1, \ldots, \widehat{\boldsymbol{P}}_T$ (the estimate of the true unknown matrix $\boldsymbol{P}_t$) for the following permanent matrix D

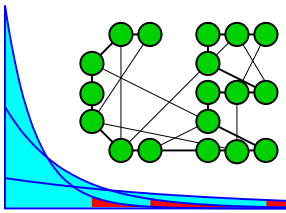


Numerical Results

The Permanent Problem: Toy Example

D:

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0

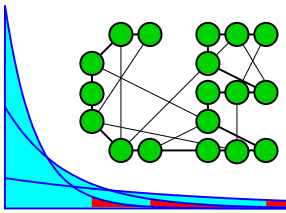


Numerical Results

The Permanent Problem

$\widehat{P}_1$

0.079	0.065	0.037	0.044	0.044	0.142	0.135	0.142	0.135	0.177
0.044	0.079	0.058	0.058	0.051	0.142	0.121	0.107	0.149	0.191
0.037	0.037	0.44	0.072	0.037	0.191	0.177	0.093	0.177	0.135
0.058	0.072	0.044	0.044	0.051	0.128	0.135	0.177	0.142	0.149
0.044	0.051	0.037	0.044	0.051	0.128	0.205	0.142	0.163	0.135
0.180	0.166	0.208	0.145	0.201	0.006	0.006	0.062	0.006	0.020
0.173	0.166	0.208	0.145	0.201	0.125	0.006	0.006	0.006	0.006
0.145	0.173	0.166	0.194	0.117	0.006	0.083	0.104	0.006	0.006
0.152	0.201	0.166	0.166	0.243	0.006	0.006	0.006	0.048	0.006
0.208	0.131	0.180	0.159	0.166	0.006	0.006	0.041	0.048	0.055

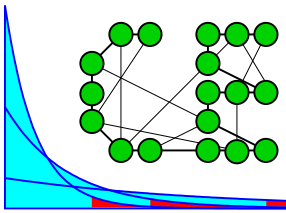


Numerical Results

The Permanent Problem

$\widehat{P}_2$

0.0237	0.0195	0.0111	0.0132	0.0132	0.1756	0.2085	0.1406	0.1945	0.2001
0.0132	0.0237	0.0174	0.0174	0.0153	0.1896	0.1343	0.1511	0.1637	0.2743
0.0111	0.0111	0.0132	0.0216	0.0111	0.2463	0.1721	0.1329	0.2631	0.1175
0.0174	0.0216	0.0132	0.0132	0.0153	0.1784	0.2435	0.1686	0.1119	0.2295
0.0132	0.0153	0.0111	0.0132	0.0153	0.1784	0.2435	0.1686	0.1119	0.2295
0.1240	0.1968	0.3214	0.1625	0.1653	0.0018	0.0018	0.0186	0.0018	0.0060
0.2829	0.1065	0.1310	0.2122	0.2227	0.0375	0.0018	0.0018	0.0018	0.0018
0.2045	0.2339	0.1338	0.2262	0.1401	0.0018	0.0249	0.0312	0.0018	0.0018
0.1716	0.2353	0.1758	0.1618	0.2339	0.0018	0.0018	0.0018	0.0144	0.0018
0.1744	0.1723	0.2080	0.1947	0.2038	0.0018	0.0018	0.0123	0.0144	0.0165

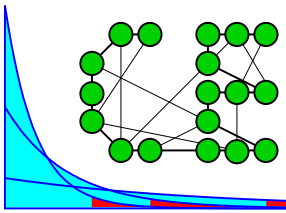


Numerical Results

The Permanent Problem

$\widehat{P}_3$:

0.0071	0.0058	0.0033	0.0039	0.0039	0.1891	0.2260	0.1421	0.1959	0.2223
0.0039	0.0071	0.0052	0.0052	0.0045	0.2015	0.1602	0.1629	0.1749	0.2740
0.0033	0.0033	0.0039	0.0064	0.0033	0.2644	0.1751	0.1539	0.2636	0.1223
0.0052	0.0064	0.0039	0.0039	0.0045	0.1423	0.1744	0.3115	0.2083	0.1391
0.0039	0.0045	0.0033	0.0039	0.0045	0.1782	0.2436	0.1988	0.1359	0.2229
0.1572	0.1708	0.2940	0.1911	0.1778	0.0005	0.0005	0.0055	0.0005	0.0018
0.2742	0.1413	0.1557	0.1824	0.2326	0.0112	0.0005	0.0005	0.0005	0.0005
0.2213	0.2289	0.1342	0.2219	0.1549	0.0005	0.0074	0.0093	0.0005	0.0005
0.1726	0.2505	0.1703	0.1708	0.2289	0.0005	0.0005	0.0005	0.0043	0.0005
0.1617	0.1916	0.2365	0.2007	0.1952	0.0005	0.0005	0.0036	0.0043	0.0049

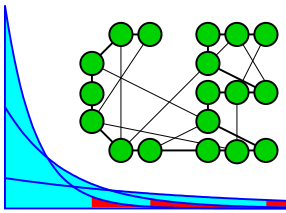


Numerical Results

The Permanent Problem

$\widehat{P}_4$:

0.0021	0.0017	0.0009	0.0011	0.0011	0.1784	0.2377	0.1570	0.1976	0.2219
0.0011	0.0021	0.0015	0.0015	0.0013	0.2017	0.1583	0.1779	0.1774	0.2766
0.0009	0.0009	0.0011	0.0019	0.0009	0.2533	0.1750	0.1687	0.2702	0.1265
0.0015	0.0019	0.0011	0.0011	0.0013	0.1635	0.1903	0.2796	0.2038	0.1552
0.0011	0.0013	0.0009	0.0011	0.0013	0.1955	0.2323	0.2074	0.1445	0.2139
0.1582	0.1786	0.2932	0.1945	0.1726	0.0001	0.0001	0.0016	0.0001	0.0005
0.2725	0.1494	0.1471	0.1780	0.2486	0.0033	0.0001	0.0001	0.0001	0.0001
0.2273	0.2149	0.1472	0.2441	0.1608	0.0001	0.0022	0.0028	0.0001	0.0001
0.1808	0.2499	0.1654	0.1786	0.2230	0.0001	0.0001	0.0001	0.0012	0.0001
0.1571	0.2020	0.2441	0.2007	0.1917	0.0001	0.0001	0.0011	0.0012	0.0014



Numerical Results

The Permanent Problem: The optimal matrix $\boldsymbol{P}_0^*$

$\boldsymbol{P}_0^*$:

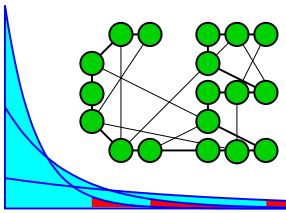
0	0	0	0	0	0.2	0.2	0.2	0.2	0.2
0	0	0	0	0	0.2	0.2	0.2	0.2	0.2
0	0	0	0	0	0.2	0.2	0.2	0.2	0.2
0	0	0	0	0	0.2	0.2	0.2	0.2	0.2
0	0	0	0	0	0.2	0.2	0.2	0.2	0.2
0.2	0.2	0.2	0.2	0.2	0	0	0	0	0
0.2	0.2	0.2	0.2	0.2	0	0	0	0	0
0.2	0.2	0.2	0.2	0.2	0	0	0	0	0
0.2	0.2	0.2	0.2	0.2	0	0	0	0	0
0.2	0.2	0.2	0.2	0.2	0	0	0	0	0

Numerical Results

The Permanent Problem: The Initial Matrix

$\boldsymbol{P}_0$:

0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.18	0.18	0.18	0.18	0.18	0.02	0.02	0.02	0.02	0.02
0.18	0.18	0.18	0.18	0.18	0.02	0.02	0.02	0.02	0.02
0.18	0.18	0.18	0.18	0.18	0.02	0.02	0.02	0.02	0.02
0.18	0.18	0.18	0.18	0.18	0.02	0.02	0.02	0.02	0.02
0.18	0.18	0.18	0.18	0.18	0.02	0.02	0.02	0.02	0.02



Numerical Results

The Permanent Problem

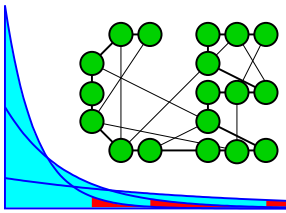
We set $\rho = 0.1,\ \alpha = 0.7,\ N = 10 * n^2$. In our tables below we list besides $|\hat{\mathcal{X}}_T^-|$, (the $(1-\rho)$-quantile of the performances) also the *best* sample performance obtained at each iteration and denoted by $|\hat{\mathcal{X}}_{t,(N)}^-|$.

To indicate the convergence of $\widehat{\boldsymbol{P}}_t$ to the optimal $\boldsymbol{P}^*$ we introduce the following quantities

$$P_t^{mm} = \min_{1\le i\le n} \max_{1\le j\le n} \widehat{p}_{t,ij}, \tag{27}$$

$t = 1, 2, \ldots$, which corresponds to the $\min\max$ value of the elements of the matrix $\widehat{\boldsymbol{P}}_{t,ij}$ at iteration t, and, similarly, the max min value

$$P_{t,mm} = \max_{1\le i\le n} \min_{1\le j\le n} \widehat{p}_{t,ij}. \tag{28}$$

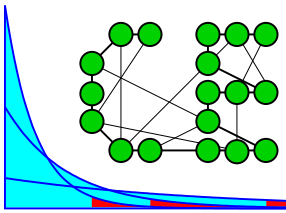


Numerical Results

The Permanent Problem

Table 1: A typical performance of Algorithm 25 for the permanent matrix D_{mn} with $m = 10$ and $n = 100$.

t	$\lvert\hat{\mathcal{X}}_t^-\rvert$	$\lvert\hat{\mathcal{X}}_{t,(N)}^-\rvert$	$\lVert\hat{\boldsymbol{P}}_t - \boldsymbol{P}^*\rVert$
1	0	5.50E-07	0.721
2	2.35E-17	4.28E-06	0.519
3	6.62E-14	0.003	0.427
4	6.14E-11	0.013	0.355
5	1.81E-08	0.131	0.296
6	2.24E-06	1.692	0.251
7	19.158	28.520	0.116
8	100	100	0.036

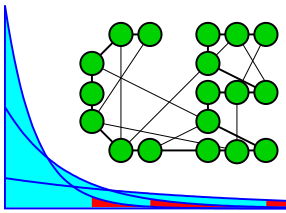


Numerical Results

The Permanent Problem

Table 2: **A typical performance of Algorithm 25 for the permanent matrix D_{mn} with $m = 10$ and $n = 100$.**

t	$\lvert\hat{\mathcal{X}}_t^-\rvert$	$\lvert\hat{\mathcal{X}}_{t,(N)}^-\rvert$	$\lvert\lvert\hat{\boldsymbol{P}}_t - \boldsymbol{P}^*\rvert\rvert$
1	0	1.664	0.199
2	1.435E-10	0.753	0.160
3	3.592E-07	1.793	0.137
4	0.052	1.825	0.130
5	11.073	18.453	0.101
6	100	100	0.067
7	100	100	0.035

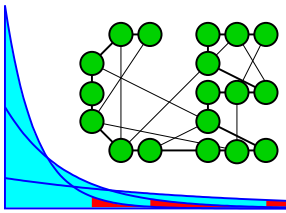


Background on MCE

The MinxEnt Program (MCE)

$$
\begin{aligned}
& \min_{f(\boldsymbol{x})} \left\{ \mathcal{D}(f|h) = \int \ln \frac{f(\boldsymbol{x})}{h(\boldsymbol{x})} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_f \ln \frac{f(\boldsymbol{X})}{h(\boldsymbol{X})} \right\} \\
(\mathrm{P}_0) \qquad & \text{s.t.} \quad \int S_j(\boldsymbol{x}) f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_f S_j(\boldsymbol{X}) = \gamma_j, \; j = 1, \ldots, k, \\
& \int f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1, \; \int h(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1.
\end{aligned}
\tag{29}
$$

Here f and h are joint n-dimensional pdf's or n-dimensional pmf's, $S_j(\boldsymbol{x}),\ j = 1, \ldots, k$, are well defined functions and $\boldsymbol{x}$ is an n-dimensional vector. The pdf h is typically assumed to be known and is called the prior pdf.



Background on MCE

The MaxEnt Program

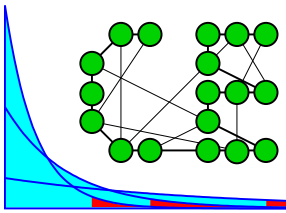
When h is **unknown**, it is taken as a uniform (continuous or discrete) pdf. In this case

$$\min_{f(\boldsymbol{x})} \left\{ \mathcal{D}(f|h) = \int \ln \frac{f(\boldsymbol{x})}{h(\boldsymbol{x})} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathbb{E}_f \ln \frac{f(\boldsymbol{x})}{h(\boldsymbol{x})} \right\} \qquad (30)$$

reduces to

$$\max_{f(\boldsymbol{x})} \left\{ \mathcal{H}(f) = - \int f(\boldsymbol{x}) \ln f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = -\mathbb{E}_f \ln f(\boldsymbol{x}) \right\}. \qquad (31)$$

The original program (P_0) is called the Kullback MinxEnt program, while the program (P_0) with (30) replaced by (31) is called the Janes MaxEnt program.



Background on MCE

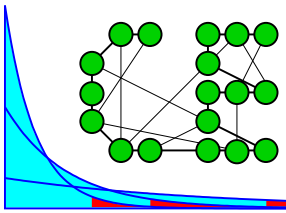
The Discrete Program:

$$\min_{\boldsymbol{p}} \mathcal{D}(\boldsymbol{p}|\boldsymbol{u}) = \min_{\boldsymbol{p}} \sum_{i=1}^{m} p_i \ln \frac{p_i}{u_i}$$

$$(\mathrm{P}_1) \qquad \textbf{s. t.} \quad \sum_{i=1}^{m} S_j(x_i) p_i = \mathbb{E}_{\boldsymbol{p}} S_j(X) = \gamma_j, \;\; j = 1, \ldots, k,$$

$$\sum_{i=1}^{m} p_i = 1, \;\; \sum_{i=1}^{m} u_i = 1, \;\; p_i \geq 0, u_i > 0. \tag{32}$$

- X denotes a random variable with m possible outcomes.
- $\boldsymbol{u} = (u_1, \ldots, u_m)$ denotes the prior probability vector of X.
- $\boldsymbol{p} = (p_1, \ldots, p_m)$ is the decision probability vector.



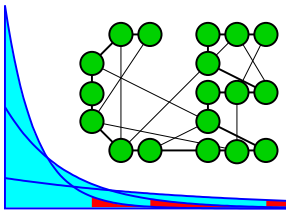
Background on MCE

The Solution of the Discrete Program (P_1)

$$
\begin{aligned}
p_i^* &= \frac{u_i \exp\{-\sum_{j=1}^k \lambda_j^* S_j(x_i)\}}{\sum_{r=1}^m u_r \exp\{-\sum_{j=1}^k \lambda_j^* S_j(x_r)\}} = \\
&= \frac{\mathbb{E}_{\boldsymbol{u}} I_{\{X=x_i\}} \exp\{-\sum_{j=1}^k \lambda_j^* S_j(X)\}}{\mathbb{E}_{\boldsymbol{u}} \exp\{-\sum_{j=1}^k \lambda_j^* S_j(X)\}}, \quad i = 1, \ldots, m,
\end{aligned} \tag{33}
$$

where $\lambda_j^*,\ j = 1, \ldots, n$ are obtained from the solution of

$$
\begin{aligned}
-\frac{\sum_{i=1}^m S_j(x_i) u_i \exp\Big\{-\sum_{r=1}^k S_r(x_i)\lambda_r\Big\}}{\sum_{i=1}^m u_i \exp\Big\{-\sum_{r=1}^k S_r(x_i)\lambda_r\Big\}} + \gamma_j &= \\
= \quad -\frac{\mathbb{E}_{\boldsymbol{u}} S_j(X) \exp\Big\{-\sum_{r=1}^k S_r(X)\lambda_r\Big\}}{\mathbb{E}_{\boldsymbol{u}} \exp\Big\{-\sum_{r=1}^k S_r(X)\lambda_r\Big\}} + \gamma_j &= 0.
\end{aligned} \tag{34}
$$



Background on MCE

A Toy Example Associated with Die Tossing

$$\min_{\boldsymbol{p}} \mathcal{D}(\boldsymbol{p}|\boldsymbol{u}) = \min_{\boldsymbol{p}} \sum_{i=1}^{6} p_i \ln \frac{p_i}{u_i}$$

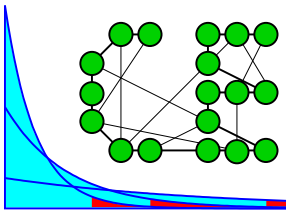
$$\text{s. t.} \quad \sum_{i=1}^{6} i p_i = \gamma, \tag{35}$$

$$\sum_{i=1}^{6} p_i = 1, \sum_{i=1}^{6} u_i = 1, \; p_i \geq 0, \; u_i > 0.$$

The solution:

$$p_i^* = \frac{u_i \exp\{-i\lambda^*\}}{\sum_{r=1}^{6} u_r \exp\{-r\lambda^*\}} = \frac{\mathbb{E}_{\boldsymbol{u}} I_{\{X=i\}} \exp\{-X\lambda^*\}}{\mathbb{E}_{\boldsymbol{u}} \exp\{-X\lambda^*\}}, \; i = 1, \ldots, 6, \tag{36}$$

and similarly λ^*.



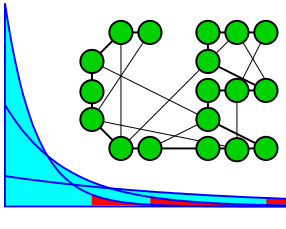
Conclusions

Advantages:

- Universally applicable (discrete/continuous/mixed problems)
- Very easy to implement
- Easy to adapt, e.g. when constraints are added
- Works generally well

Disadvantages:

- Performance function should be relatively cheap
- Tweaking (modifications) may be required



THANK YOU